5

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/725,190	11/29/2003	Douglas Wooff	50325-0842	9853

29989 7590 04/25/2007

HICKMAN PALERMO TRUONG & BECKER, LLP
2055 GATEWAY PLACE
SUITE 550
SAN JOSE, CA 95110

EXAMINER	
VU, TUAN A	
ART UNIT	PAPER NUMBER
2193	

SHORTENED STATUTORY PERIOD OF RESPONSE	MAIL DATE	DELIVERY MODE
3 MONTHS	04/25/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

If NO period for reply is specified above, the maximum statutory period will apply and will expire 6 MONTHS from the mailing date of this communication.

Office Action Summary	Application No.	Applicant(s)	
	10/725,190	WOOFF ET AL.	
	Examiner	Art Unit	
	Tuan A. Vu	2193	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *13 February 2007*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-52* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-52* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *4/18/07*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This action is responsive to the Applicant's response filed 2/13/07.

As indicated in Applicant's response, claims 1-3, 6, 10-12, 15, 19-23, 26, 30-34, 37, and 41-42 have been amended, and claims 43-52 added. Claims 1-52 are pending in the office action.

Claim Objections

2. Claim 38 is objected to because of the following informalities: It appears that the dependency of this claim to apparatus *claim 29* is a typo error, whereas mistyped *claim 29* should be apparatus *claim 32*; and the Office Action will treat this as though claim 38 would depend on claim 32.

3. The reciting of 'computer-readable medium' (see claims 20-31) entails a medium to carry signal that can be read by a computer. The *Specifications* gives examples thereof and mentions a carrier wave signal (see pg. 27-28). Should the invention contain allowable subject matter, the reciting of a potentially non-tangible computer medium would lead to a non-statutory subject matter that could be prohibitive to the allowance process. The language recited as 'readable medium' should obviate the implication of a non-tangible wave signal; e.g. be corrected to reflect a computer storage medium.

Appropriate correction is required.

Claim Rejections - 35 USC § 112

4. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

5. Claims 1, 10, 21, 32, 43 are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

The term "preferred" in those claims (i.e. *preferred software version information* – recited in the *second storage* paragraph) is a relative term which renders the claim indefinite. The term "preferred" is not defined by the claim, the specification does not provide a standard for ascertaining the requisite degree, and one of ordinary skill in the art would not be reasonably apprised of the scope of the invention. One skilled in the art would not be able gather the level of being preferred (version or a piece of information) and/or to establish a meaningful weight to this 'preferred' limitation, specially in contextual semantic (e.g. which entity is preferred over what, or with respect to whom? is it version of software or information that is preferred?); because there is no description whatsoever to set the *metes and bounds* of what characterizes the connotation of as being preferred versus not being so.

The above limitation will bear little patentable weight and will be treated as usefully stored and available version or information.

Claims 2-9, 11-20, 22-31, 33-42, 43-52 will also be rejected for no remedying to the lack of definiteness of the above relative terminology.

Claim Rejections - 35 USC § 103

6. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

7. Claims 1-52 are rejected under 35 U.S.C. 103(a) as being unpatentable over Mathur, USPN: 5,008,814 (hereinafter Mathur), in view of Kwok et al., USPN: 6,535,924 (hereinafter Kwok).

As per claim 1, Mathur discloses a method of software loading and initialization in a distributed network of nodes, the method comprising

storing, in a first storage of a master node, software packages and boot program (Fig. 3; N_S, N_o – col. 5, line 3-20; *ILP* – col 7, lines 40-64 – Note: non-volatile storage for keeping cutover software for ILP in case of need reads on storage of boot package for regressing – see col. 3 line 65 to col. 4, line 49), which software packages and boot programs will be used by the nodes in the distributed network;

storing, in a second storage of the master node, preferred software version information and node type, information for each node in the distributed network (e.g. col. 3 line 65 to col. 4, line 49; *new version, identifies node ...privileges* - col. 5, line 3-30; col. 9, line 36 to col. 10, line 8; col. 3, lines 29-53 – Note: topology information plus version and node identification in a list being stored in a distinct location for cutback after a failure in a *softload* reads on a node type and a preferred version information in a trial version to be rolled back at a second storage);

receiving, at the master node, a request for a boot program and software packages from a node, in the distributed network, that is performing an initial boot based on the request (*message receiver* – Fig. 3); retrieving preferred software version information of the node from the second storage; using the preferred software version information of the node, extracting a boot program and one or more software packages from the first storage (col. 5, line 3-30; col. 9, line 36 to col. 10, line 8; col. 7, lines 24-44 – Note: *preferred* software treated as software of

some version that has been retained for some use); delivering, to the node, the extracted boot program and one or more software packages (e.g. col. 5, line 31 to col. 6, line 54; step *202* - Fig. 2);

wherein the node stores the extracted boot program (*initial boot program* – col. 3, line 32-41) and one or more software packages in its local persistent storage;

wherein software version information is extracted from the one or more software packages and stored (e.g. col. 3 line 65 to col. 4, line 49 – Note: each node storing of a trial version at local NVRAM reads on reboots using ILP software being downloaded from the N_S master node – see Fig. 3-- with storing version thereof at the local node storage) in the local persistent storage; wherein the node reboots and executes the boot program (e.g. col. 7, lines 40-44) stored in the local persistent storage.

However, Mathur does not disclose that the boot program is a boot image; nor does Mathur disclose delivering the boot image from the master node to network node. But based on the message from a node to request a package requiring a ILP by Mathur, the need to extract boot program from a package being received for such request with which to effect an attempt to boot (see: *fails to ILP, trial use* -col. 7, lines 15-63) is strongly implied. In a multimode system with software upgrade and/or backup and topology-based routing analogous to Mathur's distribution, Kwok discloses a Master node –GMCC- storing a master image and send such image to element of group of nodes so that each node receiving such image will try to re-boot using such new image (see Fig. 4), the image including type information and programmatic for providing model components specific per node target (see col. 7, lines 32-37). Hence, it would have been obvious for one skill in the art at the time the invention was made to provide Mathur's

backup and rollback distribution with Kwok's way of providing a boot image to the network nodes from the master node. One would be motivated to do so because program components and information (e.g. node type) included in the image of the boot image --as by Kwok – would support a controllable corresponding upgrade operation for each node-specific environment, in that the image integrate therein necessary meta-information per node type (see Kwok: *Label, status* -Fig. 3; see col. 7, lines 32-37) and programmable software components needed to perform reboot including status thereof whereby to reference for future upgrade or administrative status information feedback (see Kwok: col. 7 line 61 to col. 8, line 42).

As per claim 2, Mathur discloses wherein said node, based on a command from said master node, does not store the one or more software packages in the local persistent storage device, allowing said master node to download test software packages (e.g. Fig. 2 – Note: master node transmitted version to local node reads on local node not having it stored locally prior to transmission – see step 204, step 211 – Fig. 2) to said node and temporarily run said node using the test software packages, and wherein when said node reboots, the test software packages will no longer exist on said node (step 210 – Fig. 2 – Note: trial run leading to a cutback reads on not having the bad trial software upon reboot – see col. 12, line 46-60 – where only one trial version remains after a failure and cutback).

As per claim 3, Mathur discloses wherein retrieving preferred software version information creates the preferred software version information from the second storage based on functional features requested by said node (*new version, identifies node ...privileges* - col. 5, line 3-30; col. 9, line 36 to col. 10, line 8 – Note: retrieving of proper version of boot program reads on storing of version based on stored privilege and/or identification information of nodes).

As per claim 4, Mathur discloses wherein said node verifies the software version information with said master node (step 204, step 211 – Fig. 2).

As per claim 5, Mathur discloses wherein if said node has the correct software versions, then said node completes booting by executing the software packages stored in the local persistent storage (step 211, Fig. 2; col. 3 line 65 to col. 4, line 49).

As per claim 6, Mathur discloses if said node does not have the correct software versions, retrieving correct software packages from the first storage and the correct software packages to said node (e.g. check consistency -Fig. 2), wherein said node stores the correct software packages in the local persistent storage and completes booting by executing the correct software packages stored in the local persistent storage.

As per claim 7, Mathur discloses wherein the master node has the ability to categorize nodes into classes where all of the nodes in a particular class of nodes have the same software configuration (e.g. hierarchical structure, privileges – col. 5, lines 3-27 – Note: structure representing grouping of nodes according to some particular privilege configuration reads on class of nodes of same configuration but different processor ; see col. 3, lines 29-53) and may have differing processor types.

As per claim 8, Mathur does not disclose that the master node storage of the software package contains version information, dependency information, and other metadata information pertaining to software in the package; however teaches about administrator action based on topology of common path to node for routing, and hierarchy of privileges of nodes and checksum of software destined for distribution (e.g. col 3, line 15-32; col. 5, lines 3-27; consistency check--Fig. 2); hence the concept of grouping nodes for dependency over the network topological path,

the access privileges and new software information needed would be suggestive of metadata being collected via an administrative task for providing dependency knowledge to informatively support the node distribution. Based on the boot image teaching by Kwok wherein an image would imply storing the specifics being packed for deployment (of an image) within a particular node OS environment, it would have been obvious for one skill in the art at the time the invention was made to provide the administrator or the source node– master node – with stored metadata relating to version information, dependency information as mentioned above so that this information be packed in a image as taught by Kwok; because this metadata would support the consistency checking as endeavored by Mathur in view of the topology-based for optimizing routing resources and also for identifying access privilege per nodes as mentioned above.

As per claim 9, Mathur (in view of Kwok) discloses wherein a boot image is customized for a particular type of node and provides basic low-level communications (*initial boot program* – col. 3, line 32-41; Fig. 2 – Note: checking of checksum by master node in view of software to boot the node reads on boot image having node identification and low-level instructions, because without node type specificity is provided no proper reboot would be possible).10.

As per claim 10, Mathur (in view of Kwok) discloses method of software loading and initialization in a distributed network of nodes, the method comprising

storing, in a first storage of a master node, software packages and boot images, which software packages and boot programs will be used by the nodes in the distributed network;

storing in a second storage of the master node, preferred software version information and node type, information for each node in the distributed network;

receiving, at the master node, a request for a boot images and software packages from a node, in the distributed network, that is based on the request; retrieving preferred software version information of the node from the second storage; using the preferred software version information of the node, extracting a boot images and one or more software packages from the first storage;

delivering, to the node, the extracted boot images and one or more software packages;

all of which limitations having been addressed in claim 1 (using Kwok).

As per claim 11, Mathur (in view of Kwok) discloses wherein said node stores the extracted boot image and one or more software packages in its local persistent storage (Fig. 1; col. 3 line 65 to col. 4, line 49) and wherein software version information is extracted from the one or more software packages and stored in the local persistent storage (e.g. col. 7, lines 32-53;col. 6, lines 41-54 – Note: packet reception of new software and for ILP for storage at node **reads on** extraction of package received based on version, checksum, packet time and topology of nodes etc. ; see col. 3, lines 29-53)

As per claims 12-14, refer to claims 2, 4-5 respectively.

As per claims 15-16, refer to claims 6-7, respectively.

As per claims 17-18, refer to the rationale of claim 8 and claim 9, respectively.

As per claim 19, Mathur (in view of Kwok) discloses boot images, software packages, and node information; and placing the boot programs and software packages in the first storage and the node information in the second storage on said master node (refer to claim 10); but does not explicitly disclose executing a composite image that is installed by a user onto said master node to create boot programs and packages and information. The creation of boot program and

node information being packaged for being extracted is disclosed in Mathur (re claim 10); and the administrative action to maintain version information and package for NW nodes update using operator's distribution command is taught (see *topology information, maintained* - col. 3, lines 29-53; col. 4, lined 64 to col. 5, line 27). Based on the creation of an image in Kwok's wherein package contents to support node booting are created via executing a process to generate a image, the limitation of user's installing at the master node and executing a composite program to create the image as taught in Mathur would have been obvious; that is, one of ordinary skill in the art would be motivated to provide a administrating tool by Mathur and using Kwok's teachings to support necessary files to support a programmatic content inside each boot image, provide execution by a composite process by which the necessary boot programs, packages and type information (thus persisted in Mathur's topology information) stored at the master node to create the targeted node-specific boot image as set forth in the rationale of claim 10 or claim 1.

As per claim 20, refer to claim 3.

As per claim 21, Mathur (in view of Kwok) discloses a computer-readable medium carrying one or more sequences of instructions for software loading and initialization in a distributed network of nodes, which instructions, when executed by one or more processors, cause the one or more processors to carry out the steps recited in claim 1; hence the rejection for such steps will incorporate the corresponding rejection as set forth therein respectively.

As per claims 22-27, refer to claims 11-16, respectively.

As per claim 28, refer to claim 17.

As per claims 29, 31, refer to claims 18, 20, respectively.

As per claim 30, refer to claim 19.

As per claim 32, Mathur (in view of Kwok) discloses an apparatus of software loading and initialization in a distributed network of nodes, comprising a master node and storage means with node information for supporting network node request and delivery including boot image and software packages as recited in claim 1; hence the rejection for all such limitations or means will incorporate the corresponding rejection as set forth therein respectively.

As per claims 33-38, refer to claims 22-26, respectively.

As per claim 39, refer to claim 28.

As per claims 40, 42, refer to claims 29, 31, respectively.

As per claim 41, refer to claim 30.

As per claim 43, Mathur discloses a system for software loading and initialization in a distributed network of nodes, a node in the distributed network;

a first storage on the master node, wherein the first storage stores boot programs and software packages that nodes in the distributed network will use;

a second storage on the master node, wherein the second storage stores preferred software version information and node type information for each node in the distributed network; one or more processors on the master node;

one or more sequences of instructions which, when executed by the one or more processors, cause the one or more processors to perform: receiving a request for a boot program and software packages from the node that is performing an initial boot;

based on the request, retrieving preferred software version information of the node from the second storage; using the preferred software version information of the node,

extracting a boot program and one or more software packages from the first storage; and delivering, to the node, the extracted boot program and one or more software packages;

one or more other processors on the node (e.g. Fig. 1);

one or more other sequences of instructions which, when executed by the one or more other processors, cause the one or more other processors to perform:

storing the extracted boot program and one or more software packages in local persistent storage of the node;

extracting software version information from the software packages; storing the software version information in the local persistent storage; executing the boot program, that is stored in the local persistent storage, to reboot the node;

all of which limitations being addressed in claim 1.

But Mathur does not disclose that the boot program is a boot image; nor does Mathur disclose delivering the boot image from the master node to network node. However, this boot image limitation has been addressed in claim 1using Kwok.

As per claims 44-46, refer to claims 2, 4-5, respectively.

As per claim 47, Mathur discloses wherein the one or more sequences of instructions which, when executed by the one or more processors, further cause the one or more processors to:

perform retrieving correct software packages from the first storage and sending the correct software packages to the node if the node does not have the correct software versions (re claim 6 or 15); and the one or more other sequences of instructions which, when executed by the one or more other processors, further cause the one or more other processors to

perform storing the correct software packages in the local persistent storage and executing the correct software packages stored in the local persistent storage to complete booting (re claim 6 or 15).

As per claims 48-50, refer to claims 7-9, respectively.

As per claim 51-52, refer to claims 19-20, respectively.

Response to Arguments

8. Applicant's arguments filed 2/13/07 have been fully considered but they are not persuasive. Following are the Examiner's observation in regard thereto.

35 USC § 103 Rejection:

(A) Applicants have submitted that Mathur does not teach claim 1 in terms of a node recipient that initiates the communication; i.e. no human operator required to know the specifics of a node, unlike the intervention by Mathur's human operator (Appl. Rmrks pg. 16, bottom). In order for the non-interactive paradigm as alleged above to be perceived, it is imperative that the claim be explicitly provided with teachings that clearly enforce a automatic process by which the steps as disclosed run on their own using a preconfigured code that acts like a programmatic stand-alone daemon or integrated firmware or IC, for snooping incoming requests, parsing, extracting, and making interface calls thereby collecting, compacting, and generating of a formatted data for distribution, then snoops again for more. As it stands, the steps recited as *receiving*, *extracting*, *delivering*, coupled with the subsequent actions recited as 'node … stores, … extracting … reboots' as a whole cannot enforce a unique scenario in which no human operator is implicated; nor does it clearly preclude the message received by the master node Ns in Mathur from matching claim 1 *receiving* step; nor does Mathur's network node has to be

anticipating an unclaimed feature referred to as 'initiates the communication'. Not a single teaching in the claim enforces any remote hint as to an programmatic initiation being done by a non-human entity such that the delivery/storing scenario as a whole would be as well automated necessarily. Applicant's arguments fail to comply with 37 CFR 1.111(b) because they amount to a general allegation that the claims define a patentable invention without specifically pointing out how the language of the claims patentably distinguishes them from the references.

(B) Applicants have submitted that Mathur does not teach or suggest second storage, nor does Mathur's Ns store a *preferred software version information* and node type (Appl. Rmrks pg. 17-18). In reply, the *second storage* limitation does not seem to heavily impart a patentable weight in terms of it being called *a second storage* in the scenario of the claimed steps, hence is treated as a mere storage of metadata. Consequently, any location in Mathur's master node that supports storage on network information related to a node, which is provided in the rejection, is analogized to a location is not the same as the *first location* related to storing boot or software components for packaging, i.e. otherwise recited as *first storage*. In order for a distinct second storage to bear some novel weight into the claim, more utility or action has to be recited in order to put forth the very usefulness of this storage being intentionally distinct from a first storage, e.g. being remote and structured differently from the first storage -- such that the claimed step actions will lead to a particular result, or behave in a enhanced fashion just because of the way of implementing this second storage. For lack of claim specifics about this *second storage*, this storage has been treated as a mere location in Mathur's master node to store information that is different from where the boot programs are stored. The weight (or what is left of it) of the limitation recited as 'preferred software version information' has been treated only to the extent

as set forth in USC 112 Rejection above; and accordingly, this limitation has been interpreted as though the information as recited were mere useful information to enable understanding node topology, type (i.e. *for each node in the network*), and retrieving and delivery of correct version of program to the target nodes. Applicant's arguments fail to comply with 37 CFR 1.111(b) because they amount to a general allegation that the claims define a patentable invention without specifically pointing out how the language of the claims patentably distinguishes them from the references. The argument is not convincing.

(C) Applicants have submitted that Mathur fails teach and suggest 'receiving at the master node ... boot image ... from a node'; and that it is Mathur's operator (*) that receives a boot request message, rather than a master node receiving it from a requesting node (Appl. Rmrks pg. 19). It is important to note that by just reciting 'receiving ... a request ... from a node' the claim does not create the unique interpretation that the node is equipped with API calls that automatically launch themselves without any triggering effect from another third party or external invocations. That is, when a node in a network is equipped with software that react to some events then invoke some calls to create messages for a remote service the inherency of a triggering effect caused by one event is self-evident. This event can be a network-based message invocation, a low level interrupt or catastrophic exception, or a interface event. There is no remote details provided by the claim language that the node in question would initiate a message requiring an update without any source or triggering event. It is obscure as to how this phrase recited as 'from a node' would enforce an self-triggered auto-action by a node, and thereby negate any implication of a command from a operator, as implied from the argument. One of ordinary skill in the art would interpret 'request ... from a node' as though Mathur's message

(e.g. Fig 3-8) would come from the node, and pertains to a service request destined to upgrade code of the very node, without being forced to teach that the node itself launches (by way of self-triggering) this request -- absolutely without any triggering event; rendering thereby the argument involving Mathur's operator (*) largely misplaced and un-commensurate with the language of the claim. The argument is therefore non-persuasive.

(D) Applicants have submitted that Mathur fails to teach '... retrieving preferred software version ... from the second storage' (Appl. Rmrks pg. 20). The deficient limitation recited as 'preferred software version information' has been identified and addressed according to the 35 USC § 112 rejection, and analyzed as being anticipated in section B above. The *retrieving* limitation is treated as inherent in any process included in the paradigm about delivery of components being stored from the source provider, to the target recipient; hence also disclosed. The second storage limitation has been treated as a storage to store information (node type, topology information) which distinct from the location Mathur's Ns where the boot program are persisted; and this has been fulfilled in the rejection.

(E) Applicants have submitted that Mathur fails to teach 'using the preferred ... information ... extracting ... software packages from first storage' (Appl. Rmrks pg. 21). The extracting of appropriate version of boot program in order to update a node based on a request (see Mathur: e.g. Fig 3-8; consistency check – col. 5 lines 31-46) explains that that (intrinsically and necessarily) a proper version has been retrieved; otherwise, Mathur's update process would be a long waste of resources should any random version be picked blindly and sent to the requesting node. The extracting limitation is deemed fulfilled by the scenario of node update and cutback by Mathur. The 'preferred software ... information' limitation has been addressed above. That

is, any information useful for the master node to retrieve programs related to the reboot request of a target node would be analogized as 'preferred software ...information', because dispatching of unwanted (non-preferred) version of software would inherently be very prohibitive for Mathur's distribution scheme. The argument is therefore not convincing.

(F) Applicants have submitted that Mathur does not teach 'categorize nodes ... particular class ... same configuration' that may involve different processor types; and that Mathur's cited 'privileges ... hierarchical structure' does not represent a grouping of class of nodes (Appl. Rmrks pg. 23). The limitation as recited amounts to 2 concepts: master node able to categorizing nodes into a group; the grouping as node class is being particular because the node in the class have same configuration; nothing remotely suggesting a processor type. Based on the topology teaching by Mathur wherein server retain persisting information about node, including their type and their privilege, the concept of categorizing based on a same configuration is taught. The similarly privileged group of nodes in a topology amounts to a class being particular according to such privilege configuration. The limitation as set forth in the office action is fulfilled. The language of the claim is paramount in establishing specifics that otherwise sufficiently distinguish from what it interpreted by one of ordinary skill in the art. Thus, the claim is not specific enough, and the argument is not persuasive. Applicant's arguments fail to comply with 37 CFR 1.111(b) because they amount to a general allegation that the claims define a patentable invention without specifically pointing out how the language of the claims patentably distinguishes them from the references

In all, the claims stand rejected as set forth in the Office Action.

Interview Summary

9. The Applicant's representative, Daniel Ledesma, was approached on March, 07 and in April 18, 2007 in order to establish a agreed upon grounds as to how to put certain features on the claimed Invention in a form that hopefully would enable the Invention to be more distinguishing over the prior art. But there was no agreement reached.

Conclusion

10. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tuan A Vu whose telephone number is (272) 272-3735. The examiner can normally be reached on 8AM-4:30PM/Mon-Fri.

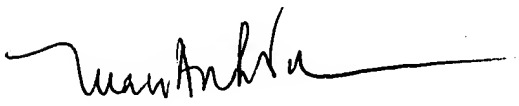
If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Meng-Ai An can be reached on (571)272-3756.

The fax phone number for the organization where this application or proceeding is assigned is (571) 273-3735 (for non-official correspondence - please consult Examiner before

using) or 571-273-8300 (for official correspondence) or redirected to customer service at 571-272-3609.

Any inquiry of a general nature or relating to the status of this application should be directed to the TC 2100 Group receptionist: 571-272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).



Tuan A Vu
Patent Examiner,
Art Unit 2193
April 20, 2007